

Scalable Dynamic Topic Modeling with Clustered Latent Dirichlet Allocation (CLDA)

Chris Gropp^{*1}, Alexander Herzog^{†1}, Ilya Safro^{‡1}, Paul W. Wilson^{§2}, and Amy W. Apon^{¶1}

¹School of Computing, Clemson University

²Department of Economics, Clemson University

October 26, 2016

Abstract

Topic modeling is an increasingly important component of Big Data analytics, enabling the sense-making of highly dynamic and diverse streams of text data. Traditional methods such as Dynamic Topic Modeling (DTM), while mathematically elegant, do not lend themselves well to direct parallelization because of dependencies from one time step to another. Data decomposition approaches that partition data across time segments and then combine results in a global view of the dynamic change of topics enable execution of topic models on much larger datasets than is possible without data decomposition. However, these methods are difficult to analyze mathematically and are relatively untested for quality of topics and performance on parallel systems. In this paper, we introduce and empirically analyze Clustered Latent Dirichlet Allocation (CLDA), a method for extracting dynamic latent topics from a collection of documents. CLDA uses a data decomposition strategy to partition data. CLDA takes advantage of parallelism, enabling fast execution for even very large datasets and a large number of topics. A large corpus is split into local segments to extract textual information from different time steps. Latent Dirichlet Allocation (LDA) is applied to infer topics at local segments. The results are merged, and clustering is used to combine topics from different segments into global topics. Results show that the perplexity is comparable and that topics generated by this algorithm are similar to those generated by DTM. In addition, CLDA is two orders of magnitude faster than existing approaches and allows for more freedom of experiment design. CLDA provides insight into how the composition of topics changes over time and can also be applied using other data partitioning strategies over any discrete features of the data, include geographic features, classes of users, and so on. In this paper CLDA is applied successfully to seventeen years of NIPS conference papers (2,484 documents and 3,280,697 words), seventeen years of computer science journal abstracts (533,560 documents and 32,551,540 words), and to forty years of the PubMed corpus (4,025,978 documents and 273,853,980 words).

^{*}cgropp@clemson.edu

[†]aherzog@clemson.edu

[‡]isafro@clemson.edu

[§]pww@clemson.edu

[¶]aapon@clemson.edu

1 Introduction

Topic modeling is a method for extracting the underlying themes from a collection of documents. One of the most common models used in practice is Latent Dirichlet Allocation (LDA) [6]. With LDA, documents are assumed to be randomly generated from one or more topics, each of which is a distribution of words. The topics are viewed as latent variables, and LDA executes by inferring the topics from the documents via a Dirichlet process. The algorithm repeatedly samples the documents and modifies the topics to better fit them until reaching a specified convergence. LDA has a number of assumptions, including that both words and documents are unordered and that all documents are generated in the same timeframe.

Of interest in streaming Big Data analytics is the modeling of topics found in a dynamic stream of data, for example, a social media data stream that changes quickly in time, or a massive collection of publications that have been produced over long time steps. Dynamic Topic Modeling (DTM) [5] relaxes the assumption of LDA that all documents are generated simultaneously. The corpus is divided into a set of time steps, each of which has its own topics. These topics represent the same theme in each time step, but the set of the most important words extracted by LDA is allowed to change through time. The mathematical foundation of DTM is well established by the authors. DTM enables observation of how the language of a topic changes over time, and also how well represented a topic is at any given point in time. However, DTM’s update process is dependent on the results of the the previous time step and is much slower than that of LDA, requiring many more iterations for convergence. Because of the dependencies, the DTM algorithm and implementation developed by Blei and Gerrish [4], is not easily parallelizable for model generation, and its application is limited to modest-sized datasets.

Typically, data decomposition approaches that partition data, say, across time segments, and then combine results in a global view enable execution on much larger datasets than is possibly without data decomposition. However, in the case of dynamic topics models this approach is difficult to analyze mathematically. Data decomposition approaches for dynamic topic modeling are relatively untested for quality of topics and performance on parallel systems. In this paper, we introduce and empirically analyze Clustered Latent Dirichlet Allocation (CLDA), a method for extracting dynamic latent topics from a collection of documents. CLDA uses a combination of LDA and clustering (in our experiments, k-means) to estimate topics that can be divided into discrete topic segments. The method has application to data streaming and can be compared to dynamic topic modeling approaches. The method can also be applied to data that can be divided according to criteria other than time such as data divided into geographic areas or with other discrete features.

CLDA executes as follows. First, the data are discretized into disjoint segments, using time steps or other criteria. Each of these segments is a sub-corpus that is used as the input to a separate run of LDA. We use PLDA+ [18], a highly parallelized implementation of LDA. Since the segments are disjoint, the runs of PLDA+ on these several segments can also be performed in parallel. The output for this step is a set of topics for every segment. The full list of these topics is passed to a parallelized implementation of k-means clustering, producing a set of topics representative of the full set. Because each step of the method uses highly parallelized code, and because the estimation of local topics can be further independently parallelized, CLDA is highly scalable and fast on even very large datasets. As a result of clustering, each original topic is a member of a particular global topic cluster, which is represented by its centroid.

CLDA has a number of promising characteristics. We show that CLDA is two orders of magnitude faster than the original implementation of DTM for data that is segmented by time steps.

In addition, analysis can be performed on the composition of each global topic in each segment, allowing a better fit at the local level than DTM. Matching the original topic mixtures to their representative centroids also enables comparison across time of global topics and analysis of how the global topics change over time. Unlike DTM, which requires that all time steps have the same number of topics, CLDA allows the number of topics to change from one timestep to another. CLDA supports analysis of data in which new topics may emerge or old topics may disappear during a time step.

In this paper, CLDA is compared empirically to the original DTM implementation with respect to the quality and similarity of the topics that are produced by the original DTM code. We choose to compare to DTM because of its strong mathematical foundations. Other methods exist, as described in Section II, but each of these varies in some way from the original DTM method. We apply CLDA successfully to seventeen years of NIPS conference papers (2,484 documents and 3,280,697 words), seventeen years of computer science journal abstracts (533,560 documents and 32,551,540 words), and to forty years of the PubMed corpus (4,025,978 documents and 273,853,980 words).

This paper has the following main contributions:

1. We introduce a new algorithm termed CLDA and demonstrate the scalability of the method. CLDA scales favorably with the number of processors, the size of the document corpus, and the number of topics across even very large datasets.
2. CLDA is analyzed with respect to the quality of the topics that are generated. CLDA is shown to compare favorably with DTM with respect to perplexity, which is the standard measurement in topic modeling for how well the model fits the data.
3. Topics produced by CLDA are compared directly to topics produced by DTM on the same corpus and number of topics. Using set-based comparison approaches we show that the topics generated by CLDA are broadly similar to those generated by DTM.
4. Implementation of CLDA is available at [1].

The remainder of this paper is structured as follows. Section II provides background information on LDA, DTM, and related approaches. Section III provides the details of the CLDA method. Section IV describes our experimental validation methodology and results. In this section, CLDA is evaluated with respect to scalability, perplexity as compared to DTM, and the similarity of the topics produced as compared to DTM. Section V discusses some of the open questions around CLDA and future work, and finally, Section VI concludes the paper.

2 Background and Related Work

2.1 Topic Modeling

Topic modeling is the process of automating the task of identifying key themes in a collection of documents. This information can be used to examine the themes in a large corpus or to classify new documents in predictive analytics. LDA [6] and is probably the most commonly used method for topic modeling. LDA assumes that words are unordered, topics are distributions of words, and multiple topics can contribute to a document that is a mixture of topics. It also assumes that

the prior distribution of each topic is a Dirichlet distribution, which distinguishes it from more generalized methods. Only the output of the model (i.e., documents) can be observed directly. The topics and topic mixtures are latent variables that must be inferred.

With the LDA method, documents are assumed to be generated by sampling a topic from their topic mixture to generate all the words in each document. Topics are randomly seeded, and then iteration proceeds using Bayesian inference. During each iteration, LDA compares each document with the topic and updates the topics for the next iteration, which continues until a stopping criteria is met. Convergence can be measured either by change in the inferred parameters, or by another objective metric of the model such as the likelihood of producing the input set [6]. Formally, for a mixture of K latent topics, where topic k is a multinomial distribution ϕ_k over a W -word vocabulary, for any document D_j , its topic mixture θ_j is a probability distribution drawn from a Dirichlet prior with parameter α . For each i^{th} word x_{ij} in d_j , a topic $z_{ij} = k$ is drawn from θ_j , and x_{ij} is drawn from ϕ_k . The generating process for LDA is

$$\theta_j \sim Dir(\alpha), \phi_k \sim Dir(\beta), z_{ij} = k \sim \phi_k. \quad (1)$$

LDA has several implementations in a variety of programming languages. There are two standard formulations, depending on how the Dirichlet priors are updated for the next iteration. The version used in the original paper and implementation uses variational Bayes, while many later works rely on Gibbs sampling [25, 15, 19, 27, 21, 14, 26]. The implementation of Gibbs sampling is less complicated and comparatively straightforward to derive.

Serial implementations of LDA do not scale well to processing of even moderately large corpora. PLDA was developed by Google Beijing Research and CMU to address the processing of very large datasets [26]. PLDA builds on a method called Approximate Distributed LDA (AD-LDA) [20]. Instead of a probability mass function, topics are stored as the count of each word assigned to them. For example, if a word is generated by a topic fifteen times and there are sixty words generated by that topic in total, AD-LDA will record fifteen whereas LDA would record 0.25. This method utilizes data parallelism by dividing up the set of documents across processes, and iterates over the corpus using Gibbs sampling. Each process has a copy of the word counts, and communicates any changes it makes to word assignment in its documents (and thus the resulting topic matrix) at the end of every iteration. During each iteration, processes do not communicate, and thus are working with stale results that are not globally accurate. As such, this is an approximation to serial Gibbs sampling. Experiments show this approximation converges in practice, and PLDA demonstrates substantial speedup on large corpora.

PLDA+ extends the implementation of PLDA and goes further by optimizing the algorithm using the four strategies of data placement, pipeline processing, word bundling, and priority-based scheduling [18]. Data placement enables the pipeline to mask communication delays with further computation, working on one word bundle while communicating the results of another. These word bundles are chosen such that the computation time is long enough to mask communication, and arranged in a circular queue rather than statically assigned to processes. The queue and word bundles are managed by one set of processors while another set performs the Gibbs sampling, thus taking advantage of model parallelism. PLDA+ succeeds in masking communication with computation, and as a result has high scalability and performance to even PLDA, which is already fast. PLDA+ nears the theoretical maximum speedup for hundreds of processes and remains very high for all process counts tested.

2.2 Dynamic Topic Modeling

One of the assumptions of LDA is that every document is equally important, but when evaluating documents over a long span of time this is problematic. For example, since language changes over time, the classification of a document written in 2000 should be based more on how it compares with documents written in the 1990s than in the 1900s. This problem can be partially sidestepped by considering blocks of time as separate collections, and performing LDA on each of them independently. This has the advantage of reducing the size of the corpus being used on any given task, which makes the method faster. But, without further processing, it has the disadvantage that it loses the information about how a topic evolves over time. CLDA addresses this weakness directly.

DTM is one approach to the time dependency problem [5]. Documents are sorted into discrete time steps, each containing a sizable corpus of its own. Each time step has its own topics, multiplying the size of the output by the number of time steps. We refer to the set of topics linked to each other over time as a *global topic*, where its representation at a given time step is a *local topic*. During each iteration, topics are updated by repeated inference on documents in their own time step, and also by consideration of the topic’s form in the preceding time step.

DTM is effective in capturing the transformation of a global topic over time. It maintains the core strength of LDA while also allowing for variance across time periods to account for slowly changing language. However, there are some weaknesses. In DTM there is not a mechanism to capture the birth or death of topics. Also, the evolution model for the topics assumes the topics are recognizable from one year to the next. While a topic might gradually evolve to be unrecognizable from its original form, each individual jump must be smaller than the distance from that topic to the others in that time.

DTM also retains the weakness of LDA to require as input the number of topics that are present. DTM adds further complication to this requirement, as the optimal number of topics may vary by time, which is not supported by the model. The time series over the data segments enables the use of Gaussian models for the time dynamics. However, the multinomial model of LDA and the Gaussian model for the time dynamics are non-conjugate, making posterior inference intractable, and an approximation must be used. Using approximations at each iteration slows the rate of convergence, causing a process already slowed by splitting the time steps to slow further so that the convergence time of DTM can be very long.

Investigation of dynamic topic modeling approaches that improves these weaknesses and increases performance is an active area of research [2, 10, 11, 8, 13, 23, 28, 17]. Table 1 provides an overview of some of the most important developments and how they compare to CLDA. Topics over time (TOT) [25] is a dynamic topic model that assumes continuous rather than discrete timestamps. iDTM [2] is an extension of DTM that relaxes the assumption that the number of topics is fixed over time, therefore allowing for the birth and death of topics. This is achieved by combining the over-time updating from DTM with a hierarchical Dirichlet process (HDP) [22], which is a model build for nested data and which has been parallelized by [9].

CLDA, described in the next section, utilizes parallel computing by applying LDA to independent data segments and combining the results using clustering. Bhadury et al. [3] devise a parallel method to address the normal complications with DTM’s inference algorithm. Previous work relies on mean field approximations, which are costly to calculate. Their work instead utilizes developments in stochastic Markov Chain Monte Carlo methods, a category which also includes Gibbs sampling. This allows them to utilize the more easily parallelized Gibbs sampling framework to estimate posterior likelihood, but is also faster in serial operation. Their results show dramatic

Table 1: Comparison of existing topic modeling approaches and CLDA

	CLDA	LDA	PLDA+	TOT	HDP	parallel HDP	DTM	iDTM	parallel DTM
Reference	[this paper]	[6]	[18]	[25]	[22]	[9]	[5]	[2]	[3]
Parallelized	✓	-	✓	-	-	✓	-	-	✓
Includes time component	✓	-	-	✓	(✓) [†]	(✓) [†]	✓	✓	✓
Evolution of topics	✓	-	-	✓	-	-	✓	✓	✓
Allows for birth/death of topics	✓	-	-	✓	-	-	-	✓	-
Unlimited number of segments	✓	-	-	✓	✓	✓	✓	✓	✓
Multiple segmentation options	✓	-	-	-	✓	✓	-	-	-

Notes:[†] HDP was built for nested data. Similar to the modeling approach presented in this paper, HDP could be applied to time-segmented data to estimate changes in topics over time.

speedup over the original DTM implementation, but the code is not available as of this writing for comparison to CLDA.

CLDA utilizes clustering in the final step of the method. Because of its speed and available code, a parallel implementation of k-means developed in [16] was used for this project. Inputs to the k-means method include the number of clusters, K , and an initial set of points. The data are classified to the nearest point using cosine similarity. There are weaknesses to this method. There is an assumption that clusters should be roughly equally sized and that data in different clusters will be separated by considerable distance. The selection of K can be problematic, and the algorithm is known to be sensitive to the initial starting points. Because of these weaknesses, exploration of other clustering approaches is a topic of future work.

3 Method Description

CLDA uses a data decomposition parallelization strategy. The data are split into multiple segments and LDA is applied to estimate local topics in each segment in parallel. Then, local topics are merged and clustering is used to calculate global topics on the merged local topics. Figure 1 provides the pseudocode and Figure 2 provides the flowchart for CLDA. The **Apply LDA** step in Figure 2 uses a plate notation [6] to illustrate the LDA method. We describe the steps of CLDA in detail.

Step 1: Split *text corpus* into S segments

First the corpus is divided into S disjoint segments on which LDA will be applied. In our application we divide the data according to naturally occurring time steps (i.e., yearly data). Other applications

```

1: procedure CLDA
2:   SPLIT text corpus into  $S$  segments
3:    $L \leftarrow$  number of local topics
4:   for all segments  $s \in \{1, \dots, S\}$  do
5:     APPLY LDA to estimate local topics  $\{t_s^i\}_{i=1}^L$ 
6:   end for
7:    $U \leftarrow$  MERGE( $\{t_s^i\}_{i=1}^L$  for  $s \in \{1, \dots, S\}$ )
8:    $K \leftarrow$  number of global topics
9:   CLUSTER  $U$  into  $K$  global topics
10: end procedure

```

▷ Lines 4-6 run independently in parallel

▷ Line 7 run in parallel

Figure 1: Pseudocode for clustered Latent Dirichlet Allocation (CLDA)

might divide data by geographical location or data source. The division of the overall corpus into individual segments can be performed as a serial task or in parallel. The vocabulary is distributed to all tasks prior to the LDA computation, and the remaining data manipulation before each LDA executes independently on the individual, smaller, segments.

LDA requires that the number of estimated topics is selected *a priori*, which we denote as L . The number of local topics L can be larger or smaller than the number of global topics K . We have found that better results are typically obtained when the number of local topics L is larger than what may be expected for global topics. The larger number of topics at the local level allows for small topics to be discovered, and for greater breadth of a topic that is unusually well represented. If many topics represent the same subject at any given segment, these are clustered together. Using more topics in the local segments avoids overfitting of the local data across the entire set of global clusters.

In this paper we describe the case where L is identical for each segment. Extensions are possible where a different L is set for each segment, either for domain-specific reasons or after determining the locally optimal number of topics through cross-validation.

Step 2: Apply LDA *to estimate local topics*

In the second step the documents in each segment are analyzed with LDA. LDA can run concurrently on separate processors (or groups of processors, if using parallel implementations of LDA such as PLDA+ in our experiments) for nearly perfect parallelism. This step results in a collection of topics $\{t_s^i\}_{i=1}^L$ at each segment $s \in \{1, \dots, S\}$, for a total of $S \cdot L$ local topics (whose merged union is denoted by U in Algorithm 1) that are clustered in the next stage.

Step 3: Merge *local topics*

The third step is to merge the emitted topics into U to obtain global topic clustering. At the conceptual level this requires concatenating the emitted topics into a single list, but in practice the step is more involved. The individual outputs $\{t_s^i\}_{i=1}^L$ have indexing entries that must be removed before they can be concatenated. The entries are then re-indexed to match the input requirements of the chosen implementation of k-means. It is also necessary to ensure that the generated topics

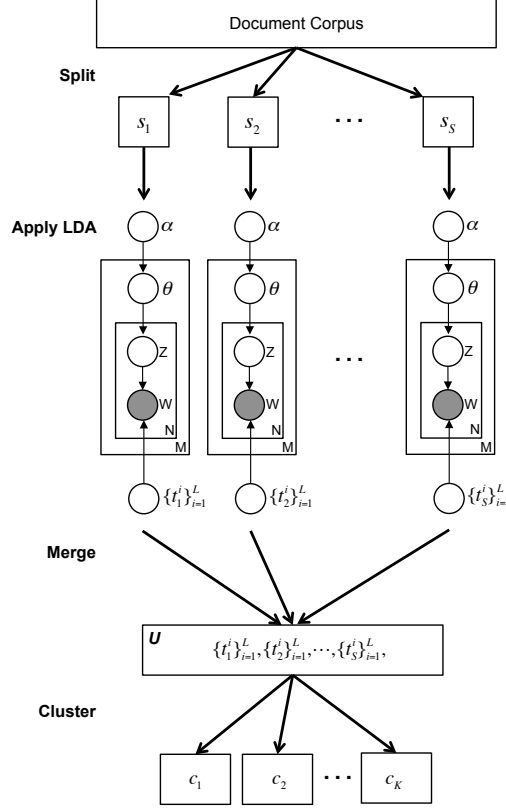


Figure 2: Flowchart of the algorithm

are comparable. LDA acts on a vocabulary consisting of everything that appears in its source documents, and produces topics with a value for each element in the vocabulary. If a word appears in one document collection but not another, these topics are not directly comparable. As such, if any of the segments does not contain the full vocabulary, it is necessary at this stage to add the missing entries to the topics, as shown in Figure 3. The entries are added with zero contribution to the topic.

In addition to ensuring the local topics are comparable in dimension, they must be comparable in scale. Some LDA implementations, including PLDA+, provide varying magnitudes for topic vectors based on their occurrence in the data. The goal of CLDA is to cluster the local topics based on the meaning, and we assume that all local topics are equally weighted. As such, the topics are normalized before clustering them. This operation is straightforward and has no dependence on other topics or other segments, and can thus be done independently before the merge, or all at once afterwards. Our implementation performs this normalization after the merge, but there is no difference in the results either way.

Step 4: Cluster *local topics*

The fourth step is to combine local topics into global topics. The k-means clustering requires that the number of global topics, which we denote as K , is set *a priori*. In the extreme cases, $K = 1$


```

1: procedure MERGE
2:   for all segments  $s \in \{1, \dots, S\}$  do
3:     for all words  $w \in W$  do
4:       if  $w \notin W_s$  then
5:         add  $w$  to  $W_s$ 
6:         for all local topic sets  $\{t_s^i\}_{i=1}^L$  do
7:            $t_s^i(w) \leftarrow 0$ 
8:         end for
9:       end if
10:    end for
11:  end for
12:   $U \leftarrow \bigcup_{s=1}^S \{t_s^i\}_{i=1}^L$ 
13: end procedure

```

▷ Lines 2-11 run independently in parallel
 ▷ W denotes full vocabulary
 ▷ W_s denotes vocabulary of s

Figure 3: Pseudocode for merging stage

defines a single cluster containing all local topics, and $K = (S \cdot L)$ defines a cluster for every topic individually. If $K > L$, not every global topic will have a representation at each segment, which means global topics will disappear and/or reappear. If $K \leq L$, topics *may* disappear and reappear at individual segments, which depends on the results of the clustering in the next stage. We consider this to be an advantage of the method over alternative implementations, such as DTM, which assumes that topics are universally represented over the entire time period.

K-means clustering is sensitive to its initialization. In CLDA, this may result in different topics. There are ways to evaluate the output of k-means across different initial values, including inter-class sum of squares, but generating initial values that are sufficiently different from each other is still a data-dependent challenge. Running LDA on the entire corpus provides a set of topics that make an intuitive set of initial values for clustering, regardless of data properties. While this can be done concurrently with running LDA on individual segments, it takes longer to complete than individual segments due to the larger corpus. To avoid the performance impact, we can use fewer iterations on the full corpus than we do on the local segments. Alternatively, instead of running LDA on the entire corpus, choose K random topics from U as the initial values. In our implementation, we run k-means on several different samplings of random initial topics, and choose the output with the best squared error. We used cosine similarity as the distance metric in our implementation. Future work will explore other metrics.

Step 5: Output *local topics and global topic assignments*

Once clustering is complete there are two important outputs. The first is the centroids themselves, each usable as a topic, and the second is the assignment of the original topics to their corresponding global clusters. The clustering outputs are useful on their own, but they provide entirely different information from the outputs of DTM. DTM does not provide a general vision of a given dynamic topic, only its local topics at each time step. CLDA provides both a segment-agnostic version of a topic and a varying number of local topics at each segment, including potentially none at all, which

Table 2: Overview of data used for evaluation

	NIPS	Computer Science Abstracts	PubMed
Time period	1987–2003	1996–2012	1976–2015
No. of segments	17	17	40
No. of documents	2,484	533,560	4,025,978
Vocabulary size	14,036	22,410	84,331
Total word tokens	3,280,697	40,002,197	273,853,980

would indicate that the topic was not meaningfully present in that segment.

4 Experimental Validation

We evaluate our method on three different data sets, which are summarized in Table 2. Our first data set is a collection of all NIPS papers from 1987 to 2003 [12], which we selected because it is a widely used data source for evaluating the quality and performance of topic models. The NIPS data contains 2,484 documents (about 150 documents per time segment), 14,036 unique words, and 3,280,697 tokens. Our second data source is a collection of abstracts from published articles in computer science provided by Elsevier and pre-processed using the open source HPCC Systems platform by LexisNexis. This data set covers the same number of time segments as the NIPS data, but includes a much larger number of documents ($N = 533,560$) with about 31,000 documents per time step. The computer science abstracts data contains 22,410 unique words and 40,002,197 tokens after removing stop words, the bottom 0.01% frequency words, and words that appeared in fewer than 0.01% of the documents. This corpus is much broader in scope and hence requires a greater number of topics to describe the documents. Our third corpus is a forty year collection of article abstracts from PubMed for the time period from 1976–2015, which contains 4,025,978 documents after we removed non-English abstracts and all articles published in journals with less than 10,000 total number of articles. The PubMed corpus contains 4,025,978 documents, 84,331 unique words and 273,853,980 tokens after removing stop words, and words that appeared less than 100 times or in fewer than 10 documents. We use this dataset to demonstrate the scalability of the approach.

4.1 Performance

We first compare CLDA’s runtime with the DTM implementation by Blei and Gerrish [4] on the computer science abstract data. These experiments use a reduced vocabulary of 1253 words, representing only words that appear in at least 1% of the documents. We execute the DTM model on 20 topics, and the CLDA model using 20 global topics and 50 local topics per segment. Blei and Gerrish’s DTM implementation is not parallelized, and PLDA+ does not run in serial, so we are unable to perfectly match the resources used. All timing experiments were run on the same hardware, only varying processor count.

The results shown in Table 3 demonstrate that the algorithm is orders of magnitude faster than the original implementation of DTM. This is unsurprising; the primary operation of consequence is the LDA phase of the algorithm, where our implementation utilizes the highly optimized PLDA+.

Table 3: Runtime Results on Computer Science Abstracts

	# of Processors	Iterations Iterations	Walltime (minutes)	Walltime (hours)
DTM	1	100	3497	58.3
CLDA	12	1,000	12	0.2
CLDA	24	1,000	6	0.1
CLDA	48	1,000	2	0.03
CLDA	48	10,000	18	0.3

The other operations largely consist of data manipulation to normalize or rotate files, and the clustering step. However, the clustering input is small compared to the size of the input data, so in our experiments k-means converged in seconds.

We next evaluate CLDA on the PubMed data, which is an order of magnitude larger than the computer science abstracts data. The LDA phase of the algorithm concluded in 22 minutes on this dataset using 1,000 iterations and 12 cores per segment. This performance takes advantage of multiple levels of parallelism, as CLDA is able to process each segment simultaneously and PLDA+ can leverage distributed computing within a segment. DTM applied to the same data with the same number of iterations would take approximately 29 weeks to complete given our earlier findings.

4.2 Quality

In order to be useful, the topics produced by the algorithm must be either very similar to those produced by DTM, or superior to them. Measuring the quality of a topic model is an open question, but a standard approximation is the perplexity metric. This metric evaluates how likely the topic model is to generate a set of provided documents. A lower perplexity indicates a model more closely fits the documents. As perplexity is a function of probabilities rather than direct model parameters, it can be used to compare different models over the same input.

Perplexity is calculated using

$$\text{perplexity} = \exp \left(- \frac{\sum_{d \in D} \sum_{w \in d} \log P(w)}{\sum_{d \in D} N_d} \right) \quad (2)$$

where d denotes a document in the corpus D , w denotes a word, and N_d denotes the number of tokens. We use a heldout set to evaluate perplexity, executing the model on 80% of the data and testing it on the remaining 20%. To evaluate the probability of generating a word $P(w)$, it is necessary to generate topic mixtures for heldout documents. We use the code provided with PLDA+ for this task [26], but a more thorough study of this problem can be found in Wallach et al.[24]

A direct comparison to DTM can be found in Table 4. These experiments use the computer science abstract data. The DTM model was executed for 58 hours using 20 topics, while the CLDA models were executed in less than an hour using $K = 20$ global topics and $L = 20$ local topics. CLDA has a comparable perplexity to Blei and Gerrish’s DTM implementation for this data set, indicating our method does not substantially degrade quality.

Table 4: Perplexity results on computer science abstracts

	Vocab. Size	Total Log- Likelihood	# of Tokens	Perplexity
DTM	22,410	-303,049,954	40,002,197	1,950
CLDA	22,410	-305,775,271	40,002,197	2,088

4.3 Similarity

The previous results indicate that our system is both very fast and has low perplexity. We wish to know how similar the generated topics are to those generated by DTM.

Topics are probability mass functions represented by vectors, but this is not how humans interpret them.[7] Rather than look holistically at the entire vector, a human will examine the most heavily weighted words in a topic; for example, the top five. These words will provide insight as to the conceptual meaning of a topic. In order to compare the insights gleaned from a set of topics, we thus need to compare what a human compares; the words most strongly tied to a topic. For a word-wise comparison of topics as sets of important words we will use the Sørensen-Dice coefficient

$$S(A, B) = \frac{2 * |A \cap B|}{|A| + |B|} \quad (3)$$

and the Jaccard index

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \quad (4)$$

In order to transform our data into a form where it can be evaluated with these metrics, we use the top 20 values in a topic as its representative set. Chang et al. [7] used the top 5 values as the core of a topic for their intruder experiment, but they were using humans to detect outliers instead of searching for broad similarity. We chose this value as it is low enough to be human-readable, but high enough to dampen the impact of minor value differences on ordering. However, this value is still arbitrary. Future work will explore other means of transforming topics into sets.

We compared the systems on two levels using both measures. We compared the global topics to each other by comparing their means. For our system, these are emitted by clustering, but for DTM we averaged the local topics together. Both the Sørensen-Dice coefficient and the Jaccard index compare single sets to each other. Comparing the outputs of DTM and CLDA requires assigning a one-to-one matching between the two collections. This is an example of the “Stable Marriage Problem” and is well studied in matching. While the general problem is NP-hard, a specific solution is possible in this case. If the topics generated by DTM and CLDA both include a topic describing the same concepts, these two topics will match more closely than they match other topics. If this is not the case, then the topics are not similar and a low value will be obtained regardless of the optimality of matching. Our experiment utilizes this assumption by greedily matching the pair of unassigned topics that are closest to each other under the Jaccard index out of all possible pairings, repeating the process until all topics are assigned. The Jaccard index and Sørensen-Dice coefficient are calculated for each match.

The values of the global matches are shown in Figure 4 sorted from best to worst. The figure shows that the closest matching topic has a 90% similarity according to the Sørensen-Dice coefficient and an 80% according to the Jaccard index. The top four to nine topics match at better than 50%.

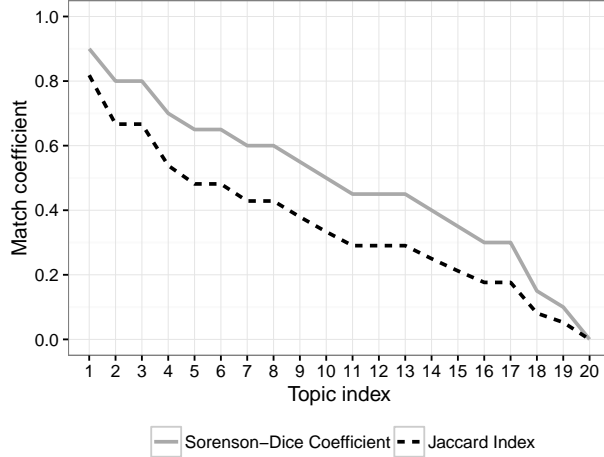


Figure 4: Similarity of global topic centroids between DTM (estimated with 20 topics) and CLDA (estimated with 20 global topics and 50 local topics) applied to the computer science abstracts data as measured by Sørensen-Dice Coefficient and Jaccard Index.

We estimate the expected value of these measures for randomly generated topics at 0.016 and 0.014, respectively, and thus conclude that CLDA and DTM are producing similar topics.

4.4 Global and local topic dynamics

LDA can be used to capture change in topic proportions over time, by executing the model over a whole corpus and then evaluating segments of it. This does not capture any change in topic language over time, forcing each segment to use the same topics. DTM relaxes this constraint by allowing the topics to vary over time. DTM produces both a version of each topic at each segment, as well as the relative proportion of each topic at each segment, demonstrating how both language and representation change over time.[5] However, DTM fixes the number of topics across time, with each overall topic having one representative per segment. CLDA relaxes this further, allowing a global topic to have any number of local representatives at each segment. In addition to allowing for topics to branch out, better fitting their local data, this also allows for global topics to appear and disappear entirely.

The strength of DTM is the variation of topics over time, taking on forms better suited to their local data while remaining tied together by a common theme. Blei et al. [5] demonstrate this by examining the changing form of a topic at several time steps, as well as their changing proportions over time. CLDA produces output to provide this same type of insight into a corpus.

We show the changing topic proportions for selected topics in both the NIPS data and computer science abstract data in Figure 5. Like DTM, CLDA provides insight into the rising and falling predominance of various topics in a corpus. Unlike DTM, CLDA global topics need not be composed of exactly one topic at each segment. Figure 6 shows how a changing number of local topics represent a global topic we identify as “Computer Networks” for six selected time segments from the computer science abstract data. While these topics are all clustered together, they represent distinct ideas within the overall concept of “Computer Networks”. One may focus on software defined networking,

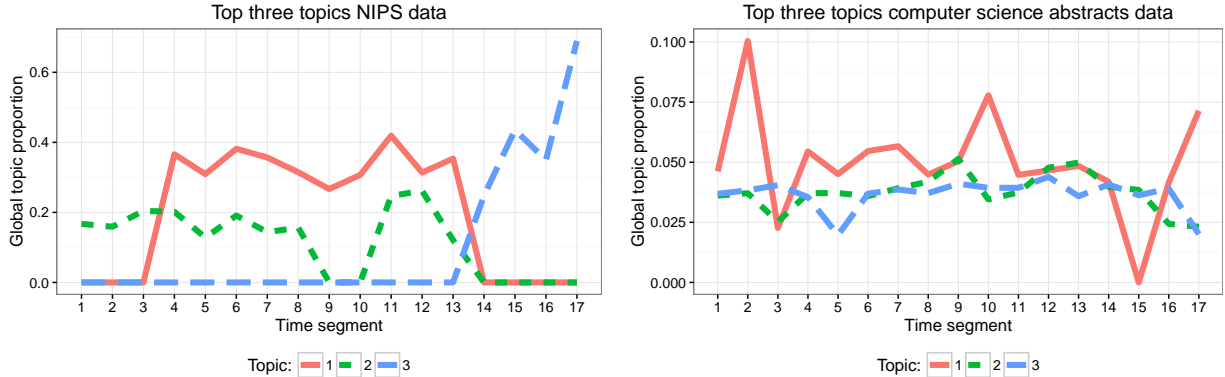


Figure 5: Evolution of three largest global topics for the NIPS data (left panel) and computer science abstracts data (right panel).

while another may focus on the communication between remote sensors. While this distinction is useful to examine, treating these as fully separate topics does not produce an accurate picture of how prevalent computer networks research is in the corpus as a whole. Clustering these topics together provides both the global insight of overall representation and local insight into a research area’s subdomains.

5 Conclusion

We have constructed and evaluated CLDA, an alternative algorithm to Dynamic Topic Modeling. This algorithm leverages existing parallel components to increase speed and facilitate the use of large corpora. It begins by discretizing the data into disjoint segments, and applying Latent Dirichlet Allocation on each segment in parallel. The resulting local topics are merged and then k-means clustering is applied, producing a number of global topics. Each global topic is composed of a number of local topics in each segment, and provides a summary of the cohesive theme across segments. Our system is built using PLDA+ and parallel k-means clustering.

We find that our system performs faster than the original implementation of DTM by two orders of magnitude. CLDA also has comparable perplexity to that of DTM. The topics generated by CLDA are similar to those generated by DTM. CLDA shows a more detailed composition of local topics than is possible with DTM, and enables global topics to emerge and disappear over the time span. Taken together, these results show that CLDA is a promising approach for modeling dynamics in topics estimated from textual data. The implementation of CLDA is available at [1].

References

- [1] Clustered LDA: implementation. Link will be added after acceptance of the paper, 2016.
- [2] Amr Ahmed and Eric P Xing. Timeline: A dynamic hierarchical dirichlet process model for recovering birth/death and evolution of topics in text stream. In *Proceedings of the 26th International Conference on Conference on Uncertainty in Artificial Intelligence*, 2010.

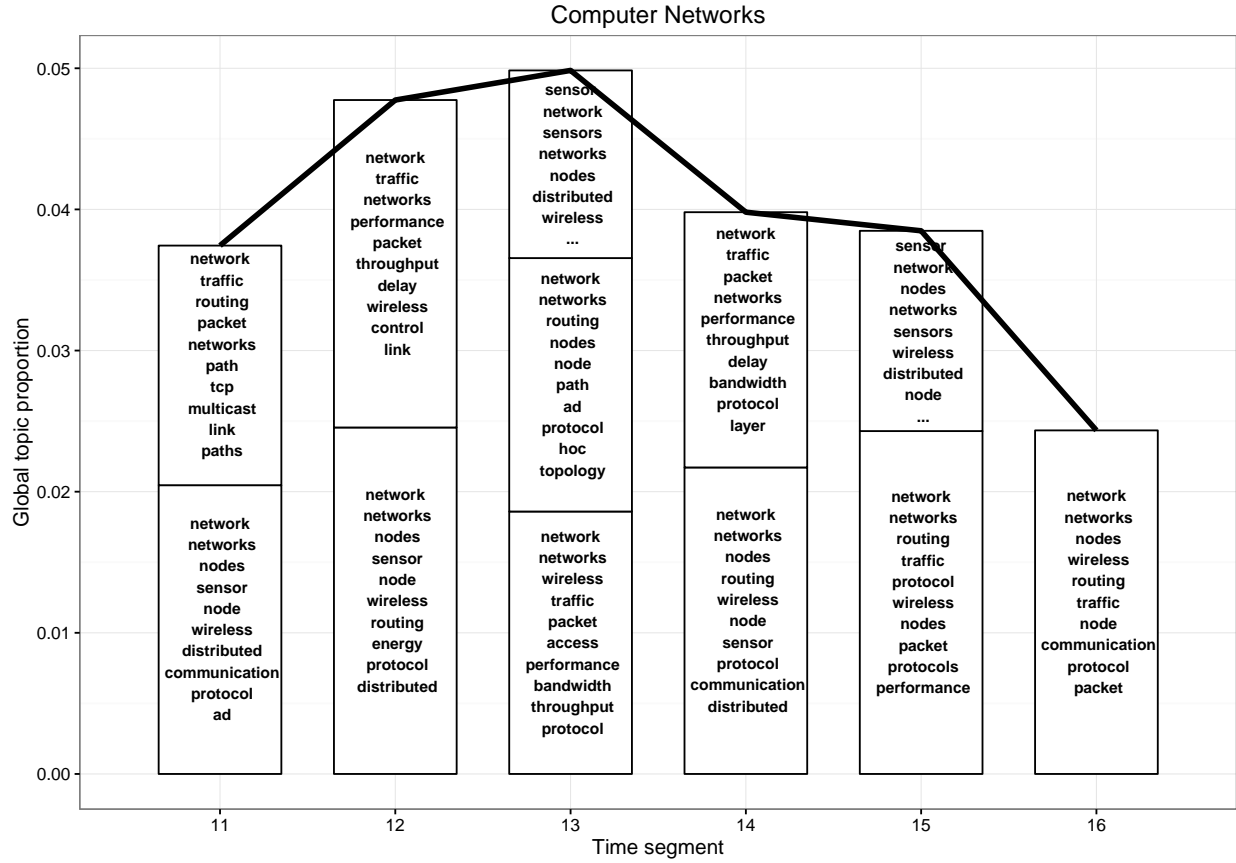


Figure 6: Local topics for selected time segments corresponding to global topic “Computer Networks” from the computer science abstracts data using 62 global topics and 50 local topics in each segment. Each bar lists the top words in each local topic. The height of each bar corresponds to the proportion a local topic contributes to the global topic.

- [3] Arnab Bhadury, Jianfei Chen, Jun Zhu, and Shixia Liu. Scaling up dynamic topic models. *arXiv preprint arXiv:1602.06049*, 2016.
- [4] David M Blei and Sean M Gerrish. Code for estimating dynamic topic models, 2011.
- [5] David M Blei and John D Lafferty. Dynamic topic models. In *Proceedings of the 23rd international conference on Machine learning*, pages 113–120. ACM, 2006.
- [6] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.
- [7] Jonathan Chang, Sean Gerrish, Chong Wang, Jordan L Boyd-Graber, and David M Blei. Reading tea leaves: How humans interpret topic models. In *Advances in neural information processing systems*, pages 288–296, 2009.
- [8] Changyou Chen, Nan Ding, and Wray Buntine. Dependent hierarchical normalized random measures for dynamic topic modeling. *arXiv preprint arXiv:1206.4671*, 2012.
- [9] Dehua Cheng and Yan Liu. Parallel gibbs sampling for hierarchical dirichlet processes via gamma processes equivalence. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 562–571. ACM, 2014.
- [10] Qiming Diao, Jing Jiang, Feida Zhu, and Ee-Peng Lim. Finding bursty topics from microblogs. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 536–544. Association for Computational Linguistics, 2012.
- [11] Avinava Dubey, Ahmed Hefny, Sinead Williamson, and Eric P Xing. A nonparametric mixture model for topic modeling over time. In *SDM*, pages 530–538. SIAM, 2013.
- [12] A. Globerson, G. Chechik, F. Pereira, and N. Tishby. Euclidean Embedding of Co-occurrence Data. *The Journal of Machine Learning Research*, 8:2265–2295, 2007.
- [13] Qi He, Bi Chen, Jian Pei, Baojun Qiu, Prasenjit Mitra, and Lee Giles. Detecting topic evolution in scientific literature: how can citations help? In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 957–966. ACM, 2009.
- [14] Yookyung Jo, John E Hopcroft, and Carl Lagoze. The web of topics: discovering the topology of topic evolution in a corpus. In *Proceedings of the 20th international conference on World wide web*, pages 257–266. ACM, 2011.
- [15] Wei Li, Xuerui Wang, and Andrew McCallum. *A continuous-time model of topic co-occurrence trends*. Defense Technical Information Center, 2006.
- [16] W-K Liao. Parallel k-means data clustering, 2013.
- [17] Kar Wai Lim and Wray L Buntine. Bibliographic analysis with the citation network topic model. In *ACML*, 2014.
- [18] Zhiyuan Liu, Yuzhou Zhang, Edward Y Chang, and Maosong Sun. Plda+: Parallel latent dirichlet allocation with data placement and pipeline processing. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):26, 2011.

- [19] Ramesh M Nallapati, Susan Dittmore, John D Lafferty, and Kin Ung. Multiscale topic tomography. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 520–529. ACM, 2007.
- [20] David Newman, Padhraic Smyth, Max Welling, and Arthur U Asuncion. Distributed inference for latent dirichlet allocation. In *Advances in neural information processing systems*, pages 1081–1088, 2007.
- [21] Yang Song, Lu Zhang, and C Lee Giles. A non-parametric approach to pairwise dynamic topic correlation detection. In *Data Mining, 2008. ICDM’08. Eighth IEEE International Conference on*, pages 1031–1036. IEEE, 2008.
- [22] Yee Whye Teh, Michael I Jordan, Matthew J Beal, and David M Blei. Hierarchical dirichlet processes. *Journal of the american statistical association*, 2012.
- [23] Yuancheng Tu, Nikhil Johri, Dan Roth, and Julia Hockenmaier. Citation author topic model in expert search. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 1265–1273. Association for Computational Linguistics, 2010.
- [24] Hanna M Wallach, Iain Murray, Ruslan Salakhutdinov, and David Mimno. Evaluation methods for topic models. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1105–1112. ACM, 2009.
- [25] Xuerui Wang and Andrew McCallum. Topics over time: a non-markov continuous-time model of topical trends. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 424–433. ACM, 2006.
- [26] Yi Wang, Hongjie Bai, Matt Stanton, Wen-Yen Chen, and Edward Y Chang. Plda: Parallel latent dirichlet allocation for large-scale applications. In *Algorithmic Aspects in Information and Management*, pages 301–314. Springer, 2009.
- [27] Xing Wei, Jimeng Sun, and Xuerui Wang. Dynamic mixture models for multiple time-series. In *IJCAI*, volume 7, pages 2909–2914, 2007.
- [28] Shuo Xu, Qingwei Shi, Xiaodong Qiao, Lijun Zhu, Hanmin Jung, Seungwoo Lee, and Sung-Pil Choi. Author-topic over time (atot): a dynamic users’ interest model. In *Mobile, Ubiquitous, and Intelligent Computing*, pages 239–245. Springer, 2014.